Attorney's Docket No.: 11333-026001

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: ESTABLISHING A VIRTUAL TUNNEL BETWEEN TWO COMPUTER PROGRAMS

APPLICANT: FELIX SHEDRINSKY

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV 382041546 US

February 20, 2004

Date of Deposit

ESTABLISHING A VIRTUAL TUNNEL BETWEEN TWO COMPUTER PROGRAMS

Cross-Reference To Related Application

This application claims priority to U.S. Provisional Application No. 60/449,213, filed on February 21, 2003, the contents of which are incorporated herein by reference.

Background

Many products, both hardware and software, have diagnostic tools or applications for communicating with them. When such products are located at customer sites, there is no way to use these tools except to send a service technician on-site to diagnose and repair any problems.

Tunneling solutions exist that provide a diagnostic program with access to a remote application. However, such tunneling solutions require a server at each site to provide access to applications behind its firewall. Since the additional server requires additional administration and maintenance at each site, it is an additional burden to end-users or customers.

Furthermore, and perhaps more significantly, installing a server at each site that provides tunnel access to a local network also has security risks. For example, such a server is addressable via the Internet, and thus is a target to break into an otherwise secure network.

Summary

In general, in one aspect, the invention is directed to a method of transferring data via a communication session between a client application and a server application. The method includes assigning an identifier to the communication session, creating at least one queue associated with the communication session, and using the identifier to store data passed between the client application and the server application in the at least one queue. The client application and the server application run local protocols, and the data is passed between the client application and the server application via an intermediary protocol. The intermediary protocol may be different from the local protocols or it may be the same protocol as the local protocols. This aspect may include one or more of the following.

A socket interface may be created to at least one of the client application and the server application. The data may be transmitted through the socket interface. The client application and the server application may be on networks that run the local protocols, and the method may include converting between the local protocols and the intermediary protocol when passing the data.

The local protocol may be TCP/IP or a serial protocol, such as RS232 and RS485. The intermediary protocol may be HTTP. The method may be performed by a server, and may also include performing load balancing to select the server from among plural servers. The identifier may be associated with the at least one queue, and may be invalidated when the communication session terminates.

The communication session may be a telnet session or may be effected via a Web site. A session record may be maintained, which includes an identity of a user initiating the session along with other information.

The method may be implemented via machine-executable instructions stored on a readable medium/media. The method may be implemented via one or more processors on one or more machines executing such instructions.

In general, in another aspect, the invention is directed to a system for transferring data via a communication session between a client application and a server application, where the client application runs on a first network and the server application runs on a second network. The system includes a proxy having a socket to the client application. The proxy converts data between a local protocol run on the first network to a non-local protocol. An agent creates a socket to the server application. The agent converts data between a local protocol run on the second network and the non-local protocol. A server is in communication with the proxy and the agent. The server contains a message queue dedicated to the communication session. The message queue stores data transmitted during the communication session. This aspect may include one or more of the following features.

The proxy may poll the server for data for the client application. When data is present for the client application, the proxy may retrieve the data from the message queue and pass the data to the client application. The agent may poll the server for data for the server application. When data is present for the client

application, the agent may retrieve the data from the message queue and pass the data to the server application.

Other features and advantages of the invention will become apparent from the following description, including the claims and drawings.

Description of the Drawings

Fig. 1 is a block diagram illustrating the concept of a virtual tunnel between two computer programs.

Fig. 2 is a block diagram of a network containing computer programs (client application and server application) that communicate via a virtual tunnel.

Fig. 3 is a flowchart showing a process for creating a virtual tunnel.

Description

The system described herein creates a virtual communication link (called a "virtual tunnel") between two computer programs (e.g., client and server applications) that are not able to address each other directly. This situation may occur when a client application needs to connect to a server application at a remote site. The server application may be on a computer on a customer or

partner's non-addressable local network (e.g., behind a firewall). As such, the client application will not be able to address the server application directly.

A virtual tunnel may be used to provide the client application access to the server application. Fig. 1 illustrates the concept of a virtual tunnel 8 between client application 10 and server application 11. Virtual tunnel 8 enables client application 10 and server application 11 to communicate as if there were no firewalls between them (which, typically, there are).

Fig. 2 shows a client local network 14. Client local network 14 includes a device 15, such as a computer, that contains a processor 16, a memory 17, and a storage medium 19 for storing, among other things, an operating system (OS) 20, a Web browser 21, software 22 for effecting network communications, and one or more executable applications (e.g., computer programs). Among these applications is client application 24. Client application 24 is a computer program for communicating with and diagnosing local or remote hardware and/or software.

A router (or modem) 25 couples client local network 14 to an external network 26, such as the Internet/World Wide

Web (Web). External network 26 may run Internet Protocol (IP), HyperText Transfer Protocol (HTTP) and other suitable protocols. Network connections may be via Ethernet, telephone line, wireless, or other transmission media.

A firewall 27 is maintained between client local network 14 and external network 26. Firewall 27 may be implemented via software run on the closest "intelligent" device to external network 26, e.g., router 25 or device 15. The firewall prevents others from directly addressing devices on client local network 14 via external network 26. As a result of the firewall, only users on client local network 14 (or some defined subset thereof) are permitted to address device 15 directly.

Client local network 14 may run a local protocol, such as Transmission Control Protocol/Internet Protocol (TCP/IP), which may be the same as, or different than, the protocols that run on external network 26. Examples of other protocols that may be run on client local network 14 include, but are not limited to, serial protocols, such as RS232 and RS485, and proprietary protocols.

Client local network 14 also includes proxy 29, which is used to effect communication between client application

24 and a remote server application. Proxy 29 may be a computer program executing on device 15 or another processing device, such as a router 25, in client local network 14. In addition to the functions described below, proxy 29 performs any conversions necessary between the protocols running on external network 26 and those running on client local network 14.

Fig. 2 also shows a server local network 30. Server local network 30 is depicted as being similar to client local network 14 for the sake of illustration. In reality, however, the two local networks may be very different.

Server local network 30 includes a device 31, such as a server, that contains a processor 32, a memory 34, and a storage medium 35 for storing, among other things, an operating system (OS) 36, software 37 for effecting network communications, and one or more executable applications (e.g., computer programs). Among these applications is server application 39. Server application 39 is a computer program that may, among other things, provide information to users via external network 26 or via local network 30. Examples of such information include, but are not limited to, Web pages and diagnostics or operational control

information pertaining to the device.

A router (or modem) 40 couples server local network 30 to external network 26. As above, network connections may be via Ethernet, telephone line, wireless, or other transmission media. A firewall 41 is also maintained between server local network 30 and external network 26. Firewall 41 may be implemented via software run on the closest "intelligent" device to external network 26, e.g., router 40 or device 31. The firewall prevents others from directly addressing device 31 via external network 26. As above, only users on server local network 30 (or some subset thereof) are permitted to address device 31 directly.

Server local network 30 may run a local protocol that may be the same as, or different than, protocols that run on external network 26 and/or client local network 14. Examples of such protocols include, but are not limited to, TCP/IP, serial protocols, such as RS232 and RS485, and proprietary protocols.

Server local network 30 also includes agent 42, which is used to effect communication between client application 24 and server application 39. Agent 42 may be a computer

program executing on device 31 or another processing device, such as a router 40, in server local network 30. In addition to the functions described below, agent 42 performs any conversions necessary between the protocols running on external network 26 and those running on server local network 30.

Agent 42 has a local configuration of interfaces. Each interface has a name, IP (Internet Protocol) address, or other information relevant to the local protocol, such as baud rate. Local configuration of the interfaces allows a local operator to control access to applications on server local network.

Proxy 29 and agent 42 perform essentially the same functions (described below) which allow client application 24 and server application 39 to communicate via a virtual tunnel. Proxy 29 and agent 42 may be pre-programmed into devices on respective local networks 14 and 30. Alternatively, one or both of proxy 29 and agent 42 may be downloaded, e.g., from external network 26. For example, in one embodiment, proxy 29 is an applet that is downloaded from a server 44 on external network 26 and that is installed on device 15. The applet may be included in a

Web page that is provided by server 44, and that is accessed by a user when establishing a virtual tunnel between client application 24 and server application 39. This process is described in more detail below.

External network 26 contains server 44, which is a computer or any other processing device. Other devices (not shown) are also located on external network 26. For example, external network 26 may contain routers, switches, and the like (not shown), which receive data packets and which forward the data packets along paths to their intended destinations. Other servers, personal computers, mainframes, and processing devices (not shown) may also be on, and/or have access to, external network 26.

Server 44 acts as an intermediary for communications between client application 24 and server application 39 in the manner described below. Server 44 runs HTTP (Hypertext Transfer Protocol) and is "visible" to other devices, such as device 15, via external network 26.

In more detail, server 44 is used in passing data between client application 24 and server application 39 because these applications cannot address each other directly. That is, since both applications are on local

networks, they do not have universally-recognized network addresses. As such, the client and server applications cannot address one another without the aid of server 44.

Server 44 includes a controller, such as a microprocessor, for executing software (machine-executable instructions) stored in a memory to perform the functions described below. To avoid confusion in terminology, the following reads as though those functions are performed by server 44, even though software in server 44, namely virtual tunneling application 45, performs the functions.

In this embodiment, server 44 contains a processor 46, a memory 47, and a storage medium 49 for storing, among other things, an operating system (OS) 50, software 51 for effecting network communication, and one or more applications. Processor 46 may execute software, including the applications, out of memory 47. Among these applications is tunneling application 45.

Tunneling application 45 creates a virtual tunnel between client application 24 on client local network 14 and server application 39 on server local network 30. Tunneling application 45 includes process 54 (Fig. 3) to establish the virtual tunnel, as described below. The

virtual tunnel allows client application 24 and server application 39 to communicate even though they are both behind firewalls and cannot directly address each other.

The virtual tunnel is also advantageous because it preserves the protocols running on local networks 14 and 30. That is, data is transferred through the virtual tunnel via an intermediary protocol, such as HTTP, that is run on external network 26. The protocols running on local networks 14 and 30 are essentially transmitted via the protocol of external network. When data reaches its destination, e.g., at local network 14 or 30, the local protocol is thus recovered by either proxy 29 or agent 42, thereby enabling the same protocol to be used at both the source and destination local networks. Proxy 29 and agent 42 are both capable of converting between a local protocol and the intermediary protocol. In this regard, it is noted that the intermediary protocol may be the same as, or different from, the local protocols.

Fig. 3 shows process 54 that is performed by proxy 29, server 44, and agent 42 to set-up a virtual tunnel for a communication session between client application 24 and server application 39. The section of Fig. 3 labeled

"Agent" corresponds to functions performed by agent 42; the section of Fig. 3 labeled "Proxy" corresponds to functions performed by proxy 29; and the section of Fig. 3 labeled "Server" corresponds to functions performed by tunneling application 45 (on server 44).

Referring to Fig. 3, agent 42 sends (60), to server 44, interface information, including, but not limited to, interface names and port numbers of agent 42 associated with server application 39. The interface information may be sent, e.g., in response to installing agent 42 on local network 30. Server 44 receives (61) the interface information and stores the interface information in a database (not shown). The database is associated with agent 42.

Information in the database may be made accessible to a user, e.g., at device 15 via a Web browser. The information may be made accessible via a Web page (not shown) provided by server 44. The Web page may contain a list of devices on local network 30 that may be accessed via virtual tunneling. The identities of the devices, which may be provided to server 44 along with the interface information, are associated with software interfaces to

such devices. When a user selects a device to connect to, the user is, in effect, selecting an interface of agent 42.

A user who wants to connect to a device on local network from client application 24 logs into server 44 via a Web page (not shown). The user then selects a device (and, thus, an interface) to begin a communication session with that device. Server 44 receives (62) the input from the Web page. As noted above, the Web page may contain an applet that comprises proxy 29. In this case, when the user selects a device and enters the selection, the applet may be installed on device 15 as proxy 29.

Server 44 may control access to agents and interfaces through a predefined security (access) policy. For example, server 44 may allow some users, but not others, access to devices on local network 30. Likewise, users may be restricted as to which devices they may access. Server 44 may control access based on user IDs (identifiers) and/or passwords assigned to system users. For example, server 44 may maintain a database of user IDs and/or passwords corresponding to devices that are accessible via those user IDs and/or passwords. If a user attempts to establish a communication session with a device for which

he has not been permitted access, server 44 may provide that user with an error message or the like.

Assuming that the user has access to the requested device, in process 54, server 44 creates (63) a session object for the current communication session. The session object contains message queues. The message queues are used to store data that is passed between client application 24 and server application 39.

In this embodiment, the session object contains two message queues (other embodiments may contain more, or less, message queues). One message queue is for data going from client application 24 to server application 39 and the other message queue is for data going from server application 39 to client application 24.

The session object also has an associated session identifier, referred to as a "sessionID" string. The sessionID string may be a unique alphanumeric identifier that identifies communications associated with a particular communication session. As described below, all data transfers associated with a communication session between client application 24 and server application 39 pass through server 44. The sessionID string is used by server

44 (in particular, by tunneling application 45) to store the data in the appropriate message queues(s).

In process 54, server 44 sends (64) the sessionID string and selected interface name to agent 42. Agent 42 receives (65) this message and creates (66) a socket using the port and IP address that correspond to the selected interface name. Agent 42 connects (67) the socket to server application 39. If connection fails, agent 42 reports an error to server 44. It is noted that sockets are used in this embodiment because they are a well-known way of communicating. Other means of communication may be used, including proxies, pipes, serial communications, etc.

In process 54, server 44 also sends (64) the sessionID string and port or other protocol parameters to proxy 29. Proxy 29 receives (68) this message and creates (69) a socket using the port that corresponds to client application 24. Proxy 29 then provides (70) an "accept" instruction on that socket. The accept instruction enables proxy 29 to accept data from client application 24.

Both proxy 29 and agent 42 execute a software thread to poll (71, 72) for data on their respective sockets, and also to poll for data from server 44. When data from

client application 24 is received on a socket of proxy 29, proxy 29 appends the appropriate sessionID string to the data and passes the data to server 44. The sessionID string enables server 44 to identify the data as belonging to a particular communication session, and to store the data in the appropriate message queue. Agent 42 polls for data from server 44, meaning that it searches for data from server 44 that is stored in message queues for agent 42. If there is data present, agent 42 retrieves the data, identifies the communication session using the sessionID string associated with the data, and passes the data to server application 39 via the established socket.

The converse occurs for data passed from server application 39 to client application 24. More specifically, data from server application 24 is received on a socket of agent 42. Agent 42 appends the sessionID string for the current communication session to the data and passes the data to server 44. Proxy 29 polls for data from server 44. If there is data present in an appropriate message queue, proxy 29 retrieves the data, identifies the communication session using the sessionID string, and passes the data to client application 24.

On each socket, a select or "recv" (receive) instruction (command) determines if there is data to read. When data is read from a socket, the data may be sent to server 44 (by proxy 29 or agent 42) as the body of an HTTP POST command. Data in server 44 may be polled (by proxy 29 or agent 42) using an HTTP GET command. If there is data in server 44, the data is passed in a reply to the GET command. This data is then written to the appropriate socket. When an HTTP command is sent, its URL parameters include a "session=SSS" parameter, where "SSS" is the sessionID for a communication session.

As noted above, server 44 has a session object that contains two message queues. Data sent from proxy 29 is stored in one message queue and delivered when agent 42 issues a GET command. The other message queue contains data being passed from agent 42 to proxy 29.

Proxy 29 and agent 42 may encrypt communications sent to server 44. Examples of encryption that may be used include Secure Sockets Layer (SSL) and Hyper Text Transfer Protocol Secure sockets (HTTPS). The recipient (e.g., proxy 29 or agent 42) should have sufficient capabilities to perform any necessary decryption.

In this embodiment, there are two ways to end a communication session (i.e., terminate a virtual tunnel). The session may end when either client application 24 or server application 39 closes its socket. However, some applications open and close sockets during the normal course of communications. For applications such as these, the user may terminate the session manually when the applications are finished running. The user may choose a termination scenario when creating the session.

Server 44 may maintain an audit log (i.e., record) of communication sessions. The audit log may identify the user, time, duration, agent, interface, and number of bytes transferred in a communication session. The actual data may be stored as human-readable text or in another format. Audit logs are particularly advantageous in diagnostic and repair scenarios, where it is often necessary to identify device modifications and repairs after the fact.

Server 44 may be associated with multiple servers, one or more of which may act as a load balancing server to distribute communications amongst other servers. In this case, when a session object is created, the session object may be created on a server that has the most (or greater

than a predetermined amount of) resources available and/or a server that is located closest to (or within a predetermined location of) agent 42. In this case, the Uniform Resource Locator (URL) of the server that is being used to effect communication is sent to agent 42 and proxy 29, along with the sessionID. In all subsequent communications, proxy 29 and agent 42 include the URL of the server. This ensures that a single server handles a single communication session.

The virtual tunnel system described herein is not limited to use with the hardware/software configuration of Figs. 2 and 3; it may find applicability in any computing or processing environment. The functionality of the virtual tunnel system, including, but not limited to, the functions performed by proxy 29, server 44, and agent 42, may be implemented in hardware (e.g., an ASIC {Application-Specific Integrated Circuit} and/or an FPGA {Field Programmable Gate Array}), software, or a combination of hardware and software.

The virtual tunnel system may find applicability in any computing or processing environment and with any type of machine that is capable of running machine-readable

instructions, such as one or more computer programs.

The virtual tunnel system can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The virtual tunnel system can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable medium or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

Method steps implemented to effect virtual tunneling can be performed by one or more programmable processors executing one or more computer programs to perform

functions described herein by operating on input data and generating output. Method steps can also be performed by, and the virtual tunnel system can be implemented as, special purpose logic circuitry.

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. Elements of a computer include a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from, or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM

and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

The virtual tunnel system can be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with the virtual tunneling system, or any combination of such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network (WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

The process described above is not limited to the implementations set forth herein. For example, the process is not limited to use with the virtual tunnel system described herein, but rather may be implemented in any type of network-based communication system.

It is noted that client application 24 and server application 39 may communicate directly when a direct connection can be achieved (e.g., if they are on the same network). To effect direct communication, if client application 24 and server application 39 use TCP/IP, server application 39 creates a socket on a specific port. Client application 24 also creates a socket and connects to this port. If server application 39 runs on another computer, client application 24 also specifies the network address of that computer. At this point, the client and server are connected and begin communicating.

The local protocols run on local networks 14, 30 may be TCP/IP or a serial protocol, such as RS232 or RS485. The protocol run on external network 26 may be HTTP. The virtual tunnel may comprise a telnet session (e.g., the tunnel is implemented during the telnet session).

It is noted that more than one agent may be present on

local network 30 and more than one proxy may be present on local network 14. There may be a one-to-one correspondence between devices and agents and between devices and proxies. Alternatively, a single proxy may service different devices and, likewise, a single agent may service different devices. Similarly, multiple proxies may service the same device and multiple agents may service the same device.

The sessionID string may expire after a predetermined period of time, necessitating a new communication session. For example, the sessionID may expire after a period during which no communications are exchanged. This period may be programmed into server 44. Similarly, the sessionID string expires when a communication session terminates.

Other embodiments not described herein are also within the scope of the following claims.

What is claimed is: